MNFRAME.005A4 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant	:	Johnson, et al.	) Group Art Unit 2785
Appl. No.	:	08/942,214	)
Filed	:	October 1, 1997	)
For	:	METHOD FOR MAPPING ENVIRONMENTAL RESOURCES TO MEMORY FOR PROGRAM ACCESS	)
Examiner	:	Norman Wright	)

DECLARATION UNDER 37 C.F.R. § 1.131 TO OVERCOME TAVALLAEI

1. This declaration is to establish the status of the invention in the above-captioned U.S. patent application in the United States on December 31, 1996, which is the effective date of U.S. Patent No. 5,864,653 entitled PCI HOT SPARE CAPABILITY FOR FAILED COMPONENTS, to Tavallaei et al., which was cited by the Examiner against the above-captioned application.

2. We are the named joint inventors of the described subject matter and all claims in the above-referenced regular patent application, filed October 1, 1997.

3. We have read the Office Actions mailed September 15, 1999, (Paper No. 13) and March 26, 2000 (Paper No. 18) regarding the patent application.

4. We reduced to practice the invention described and claimed in the pending application by at least December 19, 1996, as evidenced by the following events:

   a. By at least December 19, 1996, NetFRAME (the assignee of the subject application) was manufacturing and selling a fully functional product (the NF9000 family of network servers) that reduced to practice the claimed subject matter. The

commercial product was described as being commercially available in a document entitled "Novell IntranetWare supports hot pluggable PCI from NetFRAME," which was published on December 19, 1996, as evidenced by the document date. A copy of page 1 is attached as **Exhibit A.**

b. Reduction to practice of Claim 1. A document entitled "Raptor Wire Service Architecture, Version 1.3" (hereinafter RWSA), dated October 3, 1996, and a software source code document entitled "Module CS9000WS.SDL" (hereinafter CS9000WS), bearing a revision date of October 25, 1996, together show the invention as set out in Claim 1, and as incorporated in the product sold by NetFRAME. Copies of the cover and page 1 of RWSA are attached as **Exhibit B,** and copies of pages 1 and 5-11 of CS900WS are attached as **Exhibit C.**

For reference, Claim 1 of the pending patent application recites a "method of mapping environmental resources to memory, comprising:

providing a computer, the computer comprising a processor and a memory;
providing a microcontroller network, wherein the microcontrollers provide monitoring and control functions associated with the environmental conditions internal to the computer;
storing in the memory a unique identifier for each of the functions; and
executing commands on the microcontroller network by accessing any one of the unique identifiers.

Page 1 of RWSA depicts a "Wire Service Hardware Diagram" (a more readable version of the same diagram is presented as Fig. 5A and Fig. 5B of the patent application). RWSA shows a computer (e.g., the ISA Bus to communicate with the CPU, a PCI card, and dual CPUs on a Motherboard) and a microcontroller network (e.g., Canister Controller, CPU A Controller, Chassis Controller, etc.). RWSA illustrates that the microcontroller network is connected to the computer (e.g., the microcontroller network connects to the CPU through the System Interface and the ISA Bus), to several sensors (e.g., Chassis Controller is connected to Temperature Detector on Backplane and Temperature Detector on Motherboard; CPU B Controller is connected to CPU Thermal Fault Detector), and to environmental control components (e.g., CPU A Controller controls the speed of a fan). Thus, the microcontroller network is capable of providing functions associated with the monitoring and controlling of the environmental conditions internal to the computer.

CS9000WS shows the claimed element of storing in memory a unique identifier for each of the functions. CS900WS is a header file containing data which is stored in memory at run-time, and that provides the network address, i.e., unique identifier, for each function. For example, on page 5 CS9000WS shows a section entitled "This is [sic] the Wire Service addresses for named items" where the function for the system board fan fault (WS_SB_FANFAULT) has been assigned the unique identifier 03020300h (page 9). Hence, once a unique identifier is accessed by a CPU, it causes the execution of a command, e.g., "get fan fault", on the microcontroller network. Therefore, **Exhibit B** and **Exhibit C** together depict all of the elements of Claim 1.

Furthermore, the intended purpose of one embodiment of the invention was to provide an agent external to the microcontroller network (e.g., a remote monitoring and control program) access to the functionality of the microcontroller network without the external agent having complete knowledge of the layout and functionality of each controller in the network. The claimed subject matter as depicted in RWSA and CS9000WS, and as incorporated into the commercial product sold by NetFRAME, worked for its intended purpose.

c. Reduction to practice of Claim 2. As with Claim 1, RWSA and CS900WS show the computer, microcontroller network, and a plurality of sensors, all interconnected for the purposes of monitoring and controlling the environmental conditions internal to the computer. The claim elements of assigning a unique identifier to each sensor and of providing a model of the microcontroller network in the computer memory were reduced to practice by at least October of 1996. The subject matter of Claim 2, as reduced to practice, worked for its intended purpose.

d. Reduction to practice of Claim 20. For reference, Claim 20 recites a "method of monitoring environmental conditions in a computerized environment, the method comprising:

creating a request message which identifies one or more environmental conditions internal to the computerized environment;

sending the request message from a requestor to a microcontroller network which manages the environmental conditions;

obtaining status of the conditions identified by the request message;
creating a response message which reports the status; and
sending the response message from the microcontroller network to the requestor.

The microcontroller network described in reference to Claims 1 and 2, is capable of obtaining status of the environmental conditions identified by a request message, of creating a response to the request, and of sending the response to the requestor. By at least December 19, 1996, we reduced to practice the additional claimed elements of creating a request message identifying one or more environmental conditions, and of sending the request message from the requestor to the microcontroller network. The subject matter of Claim 20, as reduced to practice, worked for its intended purpose.

5. I, Karl S. Johnson, am listed as an inventor on a provisional Patent Application No. 60/046,397, filed May 13, 1997, which is a priority application for the subject application.

6. All acts leading to the reduction of practice were performed in the United States.

We declare that all statements made herein of our own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful, false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful, false statements may jeopardize the validity of the application or any patent resulting therefrom.

Dated: _____ By: _____
Karl Johnson

Dated: _____ By: _____
Walter Wallach

Dated: 8-11-00 By: _____
Carlton Amdahl

Dated: _____ By: _____
Ken Nguyen

S:\DOCS\UFK\UFK-1023.DOC
062800

obtaining status of the conditions identified by the request message;
creating a response message which reports the status; and
sending the response message from the microcontroller network to the requestor.

The microcontroller network described in reference to Claims 1 and 2, is capable of obtaining status of the environmental conditions identified by a request message, of creating a response to the request, and of sending the response to the requestor. By at least December 19, 1996, we reduced to practice the additional claimed elements of creating a request message identifying one or more environmental conditions, and of sending the request message from the requestor to the microcontroller network. The subject matter of Claim 20, as reduced to practice, worked for its intended purpose.

5. I, Karl S. Johnson, am listed as an inventor on a provisional Patent Application No. 60/046,397, filed May 13, 1997, which is a priority application for the subject application.

6. All acts leading to the reduction of practice were performed in the United States.

We declare that all statements made herein of our own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful, false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful, false statements may jeopardize the validity of the application or any patent resulting therefrom.

Dated: August 14, 2000 By: Karl S Johnson
Karl Johnson

Dated: _____ By: _____
Walter Wallach

Dated: _____ By: _____
Carlton Amdahl

Dated: _____ By: _____
Ken Nguyen

S:\DOCS\JFK\JFK-1023.DOC
062800

obtaining status of the conditions identified by the request message;
creating a response message which reports the status; and
sending the response message from the microcontroller network to the requestor.

The microcontroller network described in reference to Claims 1 and 2, is capable of obtaining status of the environmental conditions identified by a request message, of creating a response to the request, and of sending the response to the requestor. By at least December 19, 1996, we reduced to practice the additional claimed elements of creating a request message identifying one or more environmental conditions, and of sending the request message from the requestor to the microcontroller network. The subject matter of Claim 20, as reduced to practice, worked for its intended purpose.

5. I, Karl S. Johnson, am listed as an inventor on a provisional Patent Application No. 60/046,397, filed May 13, 1997, which is a priority application for the subject application.

6. All acts leading to the reduction of practice were performed in the United States.

We declare that all statements made herein of our own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful, false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful, false statements may jeopardize the validity of the application or any patent resulting therefrom.

Dated: _____ By: _____
Karl Johnson

Dated: 8/21/00 By: _____
Walter Wallach

Dated: _____ By: _____
Carlton Amdahl

Dated: _____ By: _____
Ken Nguyen

S:\DOCS\UFK\UFK-1023.DOC
062800

EXHIBIT A

03393902/9
DIALOG(R)File 636:Gale Group Newsletter DB(TM)

03393902 Supplier Number: 46983928 (THIS IS THE FULLTEXT)
NOVELL: Novell IntranetWare supports hot pluggable PCI from NetFRAME
M2 Presswire, pN/A
Dec 19, 1996
Language: English Record Type: Fulltext
Document Type: Newswire; Trade
Word Count: 495
TEXT:
M2 PRESSWIRE-19 December 1996-NOVELL: Novell IntranetWare supports hot pluggable PCI from NetFRAME (C)1994-96 M2 COMMUNICATIONS LTD RDATE:181296 * IntranetWare customers can add and swap PCI cards in on-line systems with minimal server downtime Novell, Inc. today announced that customers using IntranetWare, Novell's full-service Internet/intranet access platform, can take advantage of both Hot Add and Hot Swap PCI with today's NetFRAME enterprise-class network servers. The companies will continue to work closely in the future to ensure that the recently proposed PCI Hot Plug standard will deliver the level of functionality that IntranetWare and NetFRAME customers depend on.

"Hot Pluggable PCI is a key technology for continual Internet and intranet availability," said William Donahoo, senior director of product marketing at Novell. "With today's requirement for 24-hour information access, server downtime resulting from server component failure, system maintenance or hardware expansion is unacceptable. Supporting this new technology brings a new level of flexibility and fault tolerance that helps customers build business-critical intranets."

Hot Pluggable PCI technology from NetFRAME, introduced October, 1996, enables IntranetWare customers to add and swap industry standard PCI boards and device drivers, while users remain on-line greatly reducing server downtime and service disruption. The technology supports PCI-based SCSI, Ethernet, FDDI and Token Ring interface cards and device drivers. System administrators can use this functionality to both repair and expand server storage and network connectivity without having to bring down either IntranetWare or the server.

"Novell is a leader in the network operating system market," said Steve Huey, vice president of marketing at NetFRAME. "We believe Novell is well positioned to shape the future of continuous intranet computing as organizations evolve their LANs into intranets. By shipping Hot Pluggable PCI technology today, NetFRAME makes it possible for IntranetWare users to deploy continuously available server environments."

By combining IntranetWare's unique ability to load and unload device drivers without downing the server with NetFRAME's Hot Pluggable PCI technology, system administrators can add new PCI devices to a server with no user downtime. For example, if a server's network adapter fails, it can be replaced without requiring an administrator to take IntranetWare off-line or re-booting the server. When a component is replaced, the card and driver are automatically identified and configured, and the card is instantly made available as a system resource.

Founded in 1983, Novell (NASDAQ: NOVL) is the world's leading provider of network software. The company offers a wide range of network solutions for distributed network, Internet, intranet and small-business markets. Novell education and technical support programs are the most comprehensive in the network computing industry. Information about Novell's complete range of products and services can be accessed on the World Wide Web at http://www.novell.com.

EXHIBIT B

Raptor Wire Service Architecture

Version 1.3

October 3, 1996

Prepared for
NetFrame Raptor Implementation Group

by
Karl Johnson (KJ)

Introduction

"Wire Service" is the code name for the Raptor project system control, diagnostic and maintenance bus (formerly known as the CDM bus). Raptor is a completely "fly by wire" system - no switch, indicator or other control is directly connected to the function it monitors or controls. Instead, all the control and monitoring connections are made by the network of processors that comprise the "Wire Service" for the system. The processors are Microchip PIC processors and the network is a 400 kbps I^2C serial bus. A limited understanding of I^2C protocol is a prerequisite for understanding Wire Service protocols (See "The I^2C-bus and how to use it" - Philips Semiconductor, Jan 1992). Control on this bus is distributed, each processor can be either a master or a slave and can control resources on itself or any other processor on the bus.

Wire Service Hardware Block Diagram

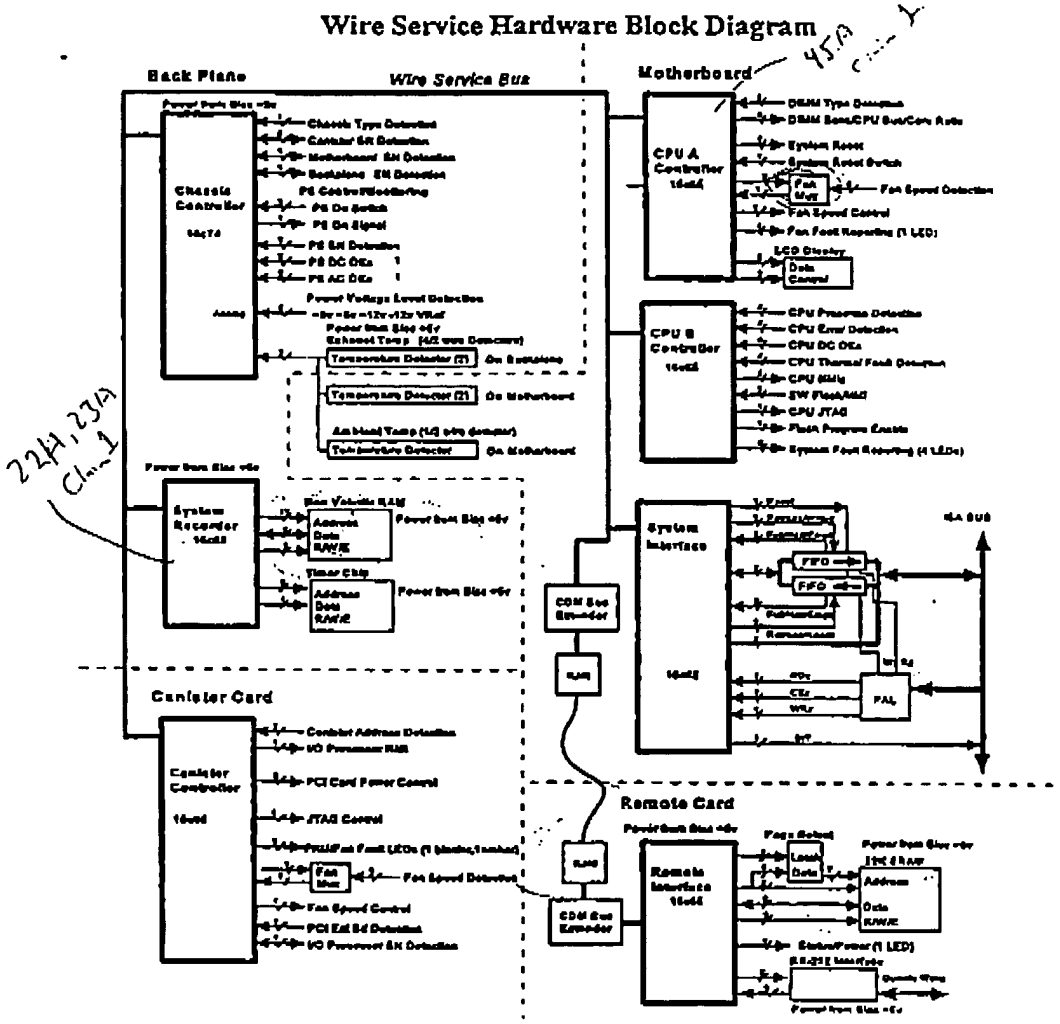

EXHIBIT C

```
;
;$Module CS9000WS.SDL$
;
;Copyright 1996
;By NetFRAME Systems Inc.
;   Milpitas, California U.S.A.
;
;$Author:   Ken Nguyen  $
;$Date:   25 Oct 1996 16:48:18  $
;$Revision
;
;$Description$
;This file contains the NetFRAME Wire Service message and interface def
inition.
; for the CS9000
;$EndDescription$
;
;  Revision History
;$Log:   P:/inc/cs9000ws.sdl  $
;
;     Rev 1.13   25 Oct 1996 16:48:18   Ken Nguyen
;  Fixed a Problem of Canister Fan Fault Status.
;
;     Rev 1.10   10 Oct 1996 16:33:04   Ken Nguyen
;  Added a command to count Log entry.
;
;     Rev 1.9   30 Sep 1996 18:42:50   Ken Nguyen
;  Added Canister Fault Commands
;
;     Rev 1.8   30 Sep 1996 17:34:16   Karl Johnson
;  Added definitions for remote interface serial protocol
;  Added NVRAM error counter
;
;     Rev 1.7   13 Sep 1996 11:22:22   Ken Nguyen
;  Corrected Temperature data length
;
;     Rev 1.6   09 Sep 1996 17:24:48   Karl Johnson
;  Added WS_SYSLOG_CLOCK - the clock used by the log recorder to time s
tamp
;
;     Rev 1.5   20 Aug 1996 01:08:36   Karl Johnson
;  Added screen event and corrected BOOTDEVS name.
;
;     Rev 1.4   01 Aug 1996 15:32:50   Karl Johnson
;  Cleanup and added new status values.
;
;     Rev 1.3   26 Jul 1996 17:14:38   Karl Johnson
;  Reduced maximum number of event types.
;  Added a Success Status.
;
;     Rev 1.2   08 Jul 1996 15:57:32   Karl Johnson
;  Changed read write bit in datatype definition.
```

```
;*************************************************************
; Wire Service Log Message Constants
;
; First byte of log message data: Severity Level Byte

WSLOG_LEVEL_UNKNOWN     CONSTANT 00h    ; Unknown
WSLOG_LEVEL_INFO        CONSTANT 10h    ; Informational
WSLOG_LEVEL_WARN        CONSTANT 20h    ; Warning
WSLOG_LEVEL_ERROR       CONSTANT 30h    ; Error
WSLOG_LEVEL_FATAL       CONSTANT 40h    ; Severe/Fatal Error


; Second byte of log message data: Source/Encoding Byte
; - which entity logged the entry in the 4 high bits
; - which type of encoding of the message is used in the 4 low bits of
the byte.
WSLOG_SRC_INTERNAL      CONSTANT 00h    ; Wire Service Internal
WSLOG_SRC_OBDIAG        CONSTANT 10h    ; Onboard Diagnostics
WSLOG_SRC_EXDIAG        CONSTANT 20h    ; External Diagnostics
WSLOG_SRC_BIOS          CONSTANT 30h    ; BIOS
WSLOG_SRC_DOS           CONSTANT 40h    ; DOS
WSLOG_SRC_WIN           CONSTANT 50h    ; Windows,Win95
WSLOG_SRC_WINNT         CONSTANT 60h    ; Windows/NT
WSLOG_SRC_NETWARE       CONSTANT 70h    ; NetWare

WSLOG_TYPE_BINARY       CONSTANT 00h    ; Message data is Binary
WSLOG_TYPE_ASCII        CONSTANT 01h    ; Message data is ASCII
WSLOG_TYPE_UNICODE      CONSTANT 02h    ; Message data is Unicode


;*************************************************************
; This is the Wire Service addresses for named items.
;
; Addresses are composed of three parts: Processor ID, Data Type and Su
baddress
; In this table the address is encoded as a 4 bytes in hexadecimal nota
tion:
; PPTTAAAAh where PP is the processor ID, TT is the data type and AL AH
 is the
; 2 byte subaddress. Processor ID's 00 and 20 are special, 00 applies t
o all
; processors and 20 applies to all canister processors.
;
;                                PPTTALAH
WS_DESCRIPTION          CONSTANT 00030100h      ;(S) Wire Service Proce
ssor Type/Description
WS_REVISION             CONSTANT 00030200h      ;(S) Wire Service Softw
are Revision/Date Info
WS_POWERUP_HOLD:        CONSTANT 01010100h      ;(L) The mode controls
action of system on A/C power available
WS_WDOG_CALLOUT         CONSTANT 01010200h      ;(L) This is a bit cont
```

```
rolling callout on a wathcdog timeout.
WS_WDOG_RESET              CONSTANT 01010300h      ;(L) This is a bit cont
rolling system on a wathcdog timeout.
WS_NVRAM_RESET             CONSTANT 01020100h      ;(B) Trigger to reset N
VRAM Data
WS_SYS_BOOTFLAG1           CONSTANT 01020200h      ;(B) System Boot Flag 1
WS_SYS_BOOTFLAG2           CONSTANT 01020300h      ;(B) System Boot Flag 2
WS_SYS_BOOTFLAG3           CONSTANT 01020400h      ;(B) System Boot Flag 3
WS_SYS_BOOTFLAG4           CONSTANT 01020500h      ;(B) System Boot Flag 4
WS_SYS_XDATA_KBYTES        CONSTANT 01020600h      ;(B) Size of the WS_SYS
_XDATA in kilobytes
WS_NVRAM_FAULTS            CONSTANT 01020700h      ;(B) Faults detected in
 NVRAM Data
WS_SCREEN_SCROLL           CONSTANT 01020800h      ;(B) Number of lines to
 scroll screen
WS_NVRAM_RES_B1            CONSTANT 01020900h      ;(B) Reserved Byte (cha
nge name to use)
WS_NVRAM_RES_B2            CONSTANT 01020a00h      ;(B) Reserved Byte (cha
nge name to use)
WS_NVRAM_RES_B3            CONSTANT 01020b00h      ;(B) Reserved Byte (cha
nge name to use)
WS_NVRAM_RES_B4            CONSTANT 01020c00h      ;(B) Reserved Byte (cha
nge name to use)
WS_NVRAM_RES_B5            CONSTANT 01020d00h      ;(B) Reserved Byte (cha
nge name to use)
WS_NVRAM_RES_B6            CONSTANT 01020e00h      ;(B) Reserved Byte (cha
nge name to use)
WS_NVRAM_RES_B7            CONSTANT 01020f00h      ;(B) Reserved Byte (cha
nge name to use)
WS_NVRAM_RES_B8            CONSTANT 01021000h      ;(B) Reserved Byte (cha
nge name to use)
WS_NVRAM_RES_B9            CONSTANT 01021100h      ;(B) Reserved Byte (cha
nge name to use)
WS_NVRAM_RES_B10           CONSTANT 01021200h      ;(B) Reserved Byte (cha
nge name to use)
WS_NVRAM_RES_B11           CONSTANT 01021300h      ;(B) Reserved Byte (cha
nge name to use)
WS_NVRAM_RES_B12           CONSTANT 01021400h      ;(B) Reserved Byte (cha
nge name to use)
WS_NVRAM_RES_B13           CONSTANT 01021500h      ;(B) Reserved Byte (cha
nge name to use)
WS_NVRAM_RES_B14           CONSTANT 01021600h      ;(B) Reserved Byte (cha
nge name to use)
WS_NVRAM_RES_B15           CONSTANT 01021700h      ;(B) Reserved Byte (cha
nge name to use)
WS_NVRAM_RES_B16           CONSTANT 01021800h      ;(B) Reserved Byte (cha
nge name to use)
WS_SYS_XDATA               CONSTANT 01070000h      ; Byte Array for storag
e of arbitrary external data in NVRAM
WS_SYS_LOG                 CONSTANT 01040000h      ; System Log
WS_RI_QUEUE                CONSTANT 01060100h      ;(Q) Queue of data goin
g to Remote Interface
```

```
WS_SI_QUEUE                 CONSTANT 01060200h      ;(Q) Queue of data goin
g to System Interface
WS_SYS_SCREEN               CONSTANT 01090000h      ; System Screen
WS_CALLOUT_SCRIPT           CONSTANT 01030300h      ;(S) The callout script
 for remote notification
WS_PASSWORD                 CONSTANT 01030400h      ;(S) The access passwor
d for Wire Service
WS_SYS_BP_SERIAL            CONSTANT 01030500h      ;(S) Last known Back Pl
ane serial data
WS_SYS_CAN_SERIAL1          CONSTANT 01030600h      ;(S) Last known Caniste
r 1 Serial data
WS_SYS_CAN_SERIAL2          CONSTANT 01030700h      ;(S) Last known Caniste
r 2 Serial data
WS_SYS_CAN_SERIAL3          CONSTANT 01030800h      ;(S) Last known Caniste
r 3 Serial data
WS_SYS_CAN_SERIAL4          CONSTANT 01030900h      ;(S) Last known Caniste
r 4 Serial data
WS_SYS_CAN_SERIAL5          CONSTANT 01030a00h      ;(S) Last known Caniste
r 5 Serial data
WS_SYS_CAN_SERIAL6          CONSTANT 01030b00h      ;(S) Last known Caniste
r 6 Serial data
WS_SYS_CAN_SERIAL7          CONSTANT 01030c00h      ;(S) Last known Caniste
r 7 Serial data
WS_SYS_CAN_SERIAL8          CONSTANT 01030d00h      ;(S) Last known Caniste
r 8 Serial data
WS_SYS_IOP_SERIAL1          CONSTANT 01030e00h      ;(S) Last known IOP in
Canister 1 Serial data
WS_SYS_IOP_SERIAL2          CONSTANT 01030f00h      ;(S) Last known IOP in
Canister 2 Serial data
WS_SYS_IOP_SERIAL3          CONSTANT 01031000h      ;(S) Last known IOP in
Canister 3 Serial data
WS_SYS_IOP_SERIAL4          CONSTANT 01031100h      ;(S) Last known IOP in
Canister 4 Serial data
WS_SYS_IOP_SERIAL5          CONSTANT 01031200h      ;(S) Last known IOP in
Canister 5 Serial data
WS_SYS_IOP_SERIAL6          CONSTANT 01031300h      ;(S) Last known IOP in
Canister 6 Serial data
WS_SYS_IOP_SERIAL7          CONSTANT 01031400h      ;(S) Last known IOP in
Canister 7 Serial data
WS_SYS_IOP_SERIAL8          CONSTANT 01031500h      ;(S) Last known IOP in
Canister 8 Serial data
WS_SYS_RI_SERIAL            CONSTANT 01031600h      ;(S) Last known Remote
Interface serial data
WS_SYS_SB_SERIAL            CONSTANT 01031700h      ;(S) Last known System
Board serial data
WS_SYS_PS_SERIAL1           CONSTANT 01031800h      ;(S) Last known Power S
upply 1 serial data
WS_SYS_PS_SERIAL2           CONSTANT 01031900h      ;(S) Last known Power S
upply 2 serial data
WS_SYS_PS_SERIAL3           CONSTANT 01031a00h      ;(S) Last known Power S
upply 3 serial data
WS_NAME                     CONSTANT 01031b00h      ;(S) System Identifying
```

```
 Name
WS_BOOTDEVS                 CONSTANT 01031c00h      ;(S) BIOS Boot drive in
formation
WS_SYS_LOG_CLOCK            CONSTANT 01031d00h      ;(S) Current time from
log timestamp clock (seconds)
WS_SYS_LOG_COUNT            CONSTANT 01031e00H      ;(S) Number of Log Entr
ies
WS_SCREEN_CURSOR_TYPE       CONSTANT 01031f00h      ;(S) Screen cursor type
 bytes (2)
WS_SCREEN_CURSOR_AT         CONSTANT 01032000h      ;(S) Screen cursor addr
ess bytes (2)
WS_SCREEN_CHANGE_INFO       CONSTANT 01032100h      ;(S) Screen change info
, Read/Only, Zero on read
WS_MODEM_INIT               CONSTANT 01032200h      ;(S) Modem initializati
on string
WS_NVRAM_RES_S1             CONSTANT 01032300h      ;(S) Reserved String (c
hange name to use)
WS_NVRAM_RES_S2             CONSTANT 01032400h      ;(S) Reserved String (c
hange name to use)
WS_NVRAM_RES_S3             CONSTANT 01032500h      ;(S) Reserved String (c
hange name to use)
WS_NVRAM_RES_S4             CONSTANT 01032600h      ;(S) Reserved String (c
hange name to use)
WS_NVRAM_RES_S5             CONSTANT 01032700h      ;(S) Reserved String (c
hange name to use)
WS_NVRAM_RES_S6             CONSTANT 01032800h      ;(S) Reserved String (c
hange name to use)
WS_NVRAM_RES_S7             CONSTANT 01032900h      ;(S) Reserved String (c
hange name to use)
WS_NVRAM_RES_S8             CONSTANT 01032a00h      ;(S) Reserved String (c
hange name to use)
WS_NVRAM_RES_S9             CONSTANT 01032b00h      ;(S) Reserved String (c
hange name to use)
WS_NVRAM_RES_S10            CONSTANT 01032c00h      ;(S) Reserved String (c
hange name to use)
WS_NVRAM_RES_S11            CONSTANT 01032d00h      ;(S) Reserved String (c
hange name to use)
WS_NVRAM_RES_S12            CONSTANT 01032e00h      ;(S) Reserved String (c
hange name to use)
WS_NVRAM_RES_S13            CONSTANT 01032f00h      ;(S) Reserved String (c
hange name to use)
WS_NVRAM_RES_S14            CONSTANT 01033000h      ;(S) Reserved String (c
hange name to use)
WS_NVRAM_RES_S15            CONSTANT 01033100h      ;(S) Reserved String (c
hange name to use)
WS_NVRAM_RES_S16            CONSTANT 01033200h      ;(S) Reserved String (c
hange name to use)
WS_SYS_POWER                CONSTANT 02010100h      ;(L) Controls system ma
ster power S4_POWER_ON
WS_SYS_REQ_POWER            CONSTANT 02010200h      ;(L) Set to request mai
n power on
WS_BP_P12V                  CONSTANT 02020100h      ;(B) Analog Measure of
```

```
+12 volt main supply
WS_BP_P3V                CONSTANT 02020200h      ;(B) Analog Measure of
+3.3 volt main supply
WS_BP_N12V               CONSTANT 02020300h      ;(B) Analog Measure of
-12 volt main supply
WS_BP_P5V                CONSTANT 02020400h      ;(B) Analog Measure of
+5 volt main supply
WS_BP_VREF               CONSTANT 02020500h      ;(B) Analog Measure of
VREF
WS_SYS_BP_TYPE           CONSTANT 02020600h      ;(B) Type of system bac
kplane currently only two types Type 0= 4 canister (small) and Type 1=
8 canister (large)
WS_SYS_CAN_PRES          CONSTANT 02020700h      ;(B) Presence bits for
canisters (LSB=1, MSB=8)
WS_SYS_PS_ACOK           CONSTANT 02020800h      ;(B) Power supply ACOK
status (LSB=1, MSB=3)
WS_SYS_PS_DCOK           CONSTANT 02020900h      ;(B) Power supply DCOK
status (LSB=1, MSB=3)
WS_SYS_PS_PRES           CONSTANT 02020a00h      ;(B) Presence bits for
power supplies (LSB=1, MSB=3)
WS_SYS_RSTIMER           CONSTANT 02020b00h      ;(B) Used to delay rese
t/run until power stabilized
WS_SYS_TEMP_SHUT         CONSTANT 02020c00h      ;(B) Shutdown temperatu
re. Initialized to ???
WS_SYS_TEMP_WARN         CONSTANT 02020d00h      ;(B) Warning temperatur
e. Initialized to ???
WS_SYS_WDOG              CONSTANT 02020e00h      ;(B) System watchdog ti
mer
WS_SYS_TEMP_DATA         CONSTANT 02030300h      ;(S) Temperatures of  a
ll sensors on temperature bus in address order
WS_SB_FAN_HI             CONSTANT 03010100h      ;(L) System Board Fans
HI
WS_SB_FAN_LED            CONSTANT 03010200h      ;(L) System Board Fan F
ault LED
WS_SYS_RUN               CONSTANT 03010300h      ;(L) Controls the syste
m halt/run line S1_OK_TO_RUN.
WS_SB_BUSCORE            CONSTANT 03020200h      ;(B) System Board BUS/C
ORE speed ratio to use on reset
WS_SB_FANFAULT           CONSTANT 03020300h      ;(B) System Board Fan f
ault bits
WS_SB_FAN_LOLIM          CONSTANT 03020400h      ;(B) Fan speed low spee
d fault limit
WS_SB_LCD_COMMAND        CONSTANT 03020500h      ;(B) Low level LCD Cont
roller Command
WS_SB_LCD_DATA           CONSTANT 03020600h      ;(B) Low level LCD Cont
roller Data
WS_SB_DIMM_TYPE          CONSTANT 03030300h      ;(S) The type of DIMM i
n each DIMM socket as a 16 byte string
WS_SB_FAN_DATA           CONSTANT 03030400h      ;(S) System Board Fan s
peed data in fan number order
WS_SYS_LCD1              CONSTANT 03030500h      ;(S) Value to display o
n LCD Top line
```


Ca

```
WS_SYS_LCD2                CONSTANT 03030600h      ;(S) Value to display o
n LCD Bottom line
WS_SB_LCD_STRING           CONSTANT 03030700h      ;(S) Low Level LCD Disp
lay string at current position
WS_NMI_REQ                 CONSTANT 04010100h      ;(L) NMI Request bit
WS_SB_CPU_FAULT            CONSTANT 04010200h      ;(L) CPU Fault Summary
WS_SB_FLASH_ENA            CONSTANT 04010300h      ;(L) Indicates FLASH RO
W write enabled
WS_SB_FRU_FAULT            CONSTANT 04010400h      ;(L) Indicates the FRU
status
WS_SB_JTAG                 CONSTANT 04010500h      ;(L) Enables JTAG chain
 on system board
WS_SYSFAULT                CONSTANT 04010600h      ;(L) System Fault Summa
ry
WS_SYS_OVERTEMP            CONSTANT 04010700h      ;(L) Indicates Overtemp
 fault
WS_CAN1_FAN_SYSFLT         CONSTANT 04010800H      ;(L) Indicates Canister
 #1 Fan System Fault
WS_CAN2_FAN_SYSFLT         CONSTANT 04010900H      ;(L) Indicates Canister
 #2 Fan System Fault
WS_CAN3_FAN_SYSFLT         CONSTANT 04010A00H      ;(L) Indicates Canister
 #3 Fan System Fault
WS_CAN4_FAN_SYSFLT         CONSTANT 04010B00H      ;(L) Indicates Canister
 #4 Fan System Fault
WS_CAN5_FAN_SYSFLT         CONSTANT 04010C00H      ;(L) Indicates Canister
 #5 Fan System Fault
WS_CAN6_FAN_SYSFLT         CONSTANT 04010D00H      ;(L) Indicates Canister
 #6 Fan System Fault
WS_CAN7_FAN_SYSFLT         CONSTANT 04010E00H      ;(L) Indicates Canister
 #7 Fan System Fault
WS_CAN8_FAN_SYSFLT         CONSTANT 04010F00H      ;(L) Indicates Canister
 #8 Fan System Fault
WS_NMI_MASK                CONSTANT 04020100h      ;(B) CPU NMI processor
mask (LSB=CPU1)
WS_SB_CPU_ERR              CONSTANT 04020200h      ;(B) CPU Error bits (LS
B = CPU1)
WS_SB_CPU_POK              CONSTANT 04020300h      ;(B) CPU Power OK (LSB
= CPU1)
WS_SB_CPU_PRES             CONSTANT 04020400h      ;(B) CPU Presence bits
(LSB = CPU1)
WS_SB_CPU_TEMP             CONSTANT 04020500h      ;(B) CPU Thermal fault
bits (LSB = CPU1)
WS_SI_EVENTS               CONSTANT 10050100h      ;(E) System Interface E
vent Queue
WS_RI_CD                   CONSTANT 11010100h      ;(L) Status of Remote P
ort Modem CD
WS_RI_CTS                  CONSTANT 11010200h      ;(L) Status of Remote P
ort Modem CTS
WS_RI_DSR                  CONSTANT 11010300h      ;(L) Status of Remote P
ort Modem DSR
WS_RI_DTR                  CONSTANT 11010400h      ;(L) State of Remote Po
rt Modem DTR
```

```
WS_RI_RTS               CONSTANT 11010500h      ;(L) Status of Remote P
ort Modem RTS
WS_RI_CALLOUT           CONSTANT 11020100h      ;(B) Controls Call out
Script activation
WS_RI_EVENTS            CONSTANT 11050100h      ;(E) Remote Interface E
vent Queue
WS_CAN_FAN_HI           CONSTANT 20010100h      ;(L) Canister Fans HI
WS_CAN_FAN_LED          CONSTANT 20010200h      ;(L) Canister Fan Fault
 LED
WS_CAN_JTAG_ENA         CONSTANT 20010300h      ;(L) Enable JTAG TMS ch
ain for canister
WS_CAN_NMI_S5           CONSTANT 20010400h      ;(L) NMI card in slot 5
WS_CAN_POWER            CONSTANT 20010500h      ;(L) Controls canister
PCI slot power
WS_CAN_S5_PRESENT       CONSTANT 20010600h      ;(L) Indicates the pres
ence of something in slot 5
WS_CAN_S5_SMART         CONSTANT 20010700h      ;(L) Indicates somethin
g other than a passive board in slot 5
WS_CAN_FAN_LOLIM        CONSTANT 20020100h      ;(B) Fan low speed faul
t limit
WS_CAN_PCI_PRESENT      CONSTANT 20020200h      ;(B) Reflects PCI card
slot[1..4] presence indicator pins ( MSB to LSB) 4B,4A,3B,3A,2B,2A,1B,1
A
WS_CAN_FANFAULT         CONSTANT 20020300h      ;(B) Canister Fan Fault
 Bits
WS_CAN_FAN_DATA         CONSTANT 20030300h      ;(S) Canister Fan speed
 data


;*********************************************************
; This is the Wire Service Attributes for named items.
; The attribute information is stored in a symbolic constant named the
same
; as the named item then followed by two underscores
;
; Attributes consist of:
;       R/W access for internal WS (I), BIOS/OS (O), administrator (A),
 and general (G)
;       groups. ( 0 = NoAccess 1 = Read Only, 2 = Write Only, 3 = Read/
Write )
;
;       maximum possible reques/response length of item in bytes (LL)
;
;       Group Name ID (ID)
;
;                                IOAGLLID
WS_DESCRIPTION__        CONSTANT 11114000h      ;(S) Wire Service Proce
ssor Type/Description
WS_REVISION__           CONSTANT 11112000h      ;(S) Wire Service Softw
are Revision/Date Info
WS_POWERUP_HOLD__       CONSTANT 33310100h      ;(L) The mode controls
action of system on A/C power available
```